下一代互联网技术

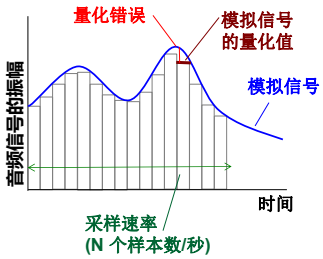
第五章　多媒体网络

第五章 多媒体网络

- 多媒体网络应用
- 流式存储视频
- IP语音
- 实时会话式应用的协议
- 支持多媒体的网络

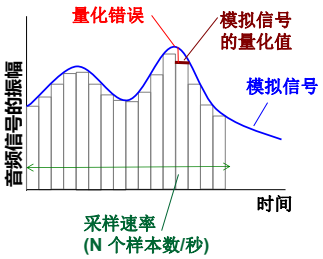
多媒体音频

- 按恒定速率采样的模拟音频信号
  - 电话: 8,000 个样本/秒
  - CD音乐: 44 100 个样本/秒
- 每个采样值被量化，例如："四舍五入"
  - 量化值通常是2的幂，例如 2^8=256个量化值
  - 每个量化值用若干比特来表示，例如：256个量化值，每个值用8 bits表示



多媒体音频

- 例子: 以每秒8,000个样本采样，每个样本被量以8比特表示，则得到的数字信号速率为64,000比特/秒
- 接收器将比特转换为模拟信号:
  - 有少许质量下降

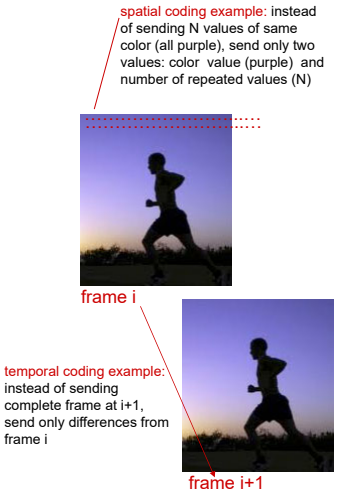


速率实例:

- CD: 1.411 Mbps
- MP3: 96, 128, 160 kbps
- 因特网电话: 5.3 kbps 以上

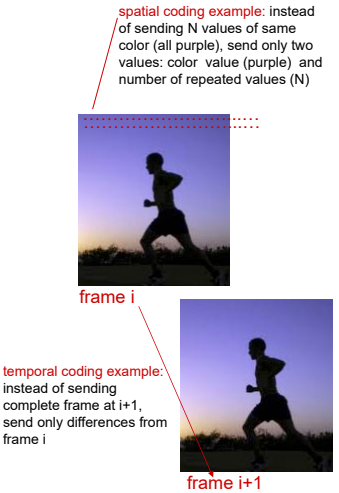
多媒体视频

- 视频: 按恒定速率显示的图像序列，例如：24幅图像/秒
- 数字编码图像: 由像素阵列组成
  - 每个像素被编码为一定数量的比特来表示亮度和颜色
- 编码: 利用图像冗余可以进行视频压缩，减少图像编码后的大小。
  - 空间冗余 (图像内部的空白)
  - 时域冗余 (图像与后续图像的重复程度)



多媒体视频

- *CBR: (constant bit rate):* 视频编码速率固定
- *VBR: (variable bit rate):* 视频编码率随着空间、时间编码的变化而变化
- 例子:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (通常用于因特网, < 1 Mbps)



多媒体网络3种应用类型

- **流式存储音/视频**
  - **流: 开始从服务器接收文件几秒后就可以播放，同时继续从服务器接收该视频后续部分。**
  - **存储 (在服务器): 客户端缓存的应用使得**
  - **例如：YouTube, Netflix, 优酷**
- **会话式IP语音/视频**
  - **类似人类对话的交互行为，是典型的时延敏感和容忍丢包的应用。**
  - **例如Skype**
- **流式实况音/视频**
  - **例如：因特网电台广播、实况体育事件**
  - **许多技术类似于流式存储音/视频**

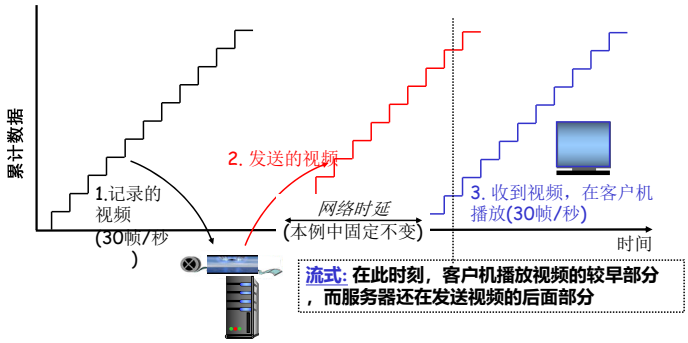
第五章 多媒体网络

- **多媒体网络应用**
- **流式存储视频**
- **IP语音**
- **实时会话式应用的协议**
- **支持多媒体的网络**

流式存储多媒体

- **对于流式视频应用，预先录制的视频放置在服务器上，用户向这些服务器发送请求，按需、交互式的观看视频。**
- **使用客户端应用缓存，以缓解变化的端到端时延和变化的可用带宽的影响**
- **流式视频系统分为三种类型:**
  - **UDP流**
  - **HTTP流**
  - **适应性HTTP流**

流式存储多媒体

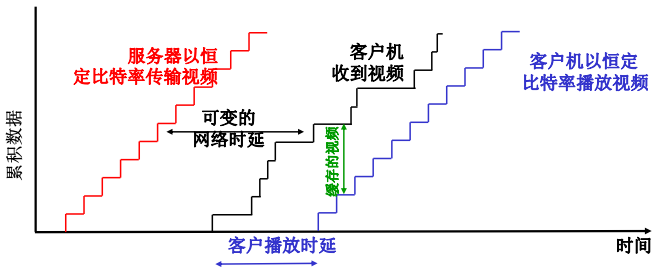


客户端交互：暂停，快进，快退，视频跳转

流式存储多媒体

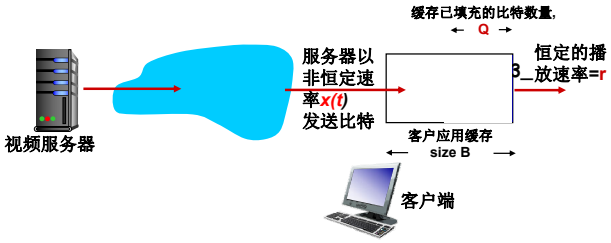
- **连续播放的约束: 一旦客户端开始播放视频，应该匹配该视频初始记录的时序。**
  - **… 但由于网络时延可变（抖动），需要客户端缓存以匹配连续播放的约束**
- **其他挑战:**
  - **客户端交互：暂停、快进、快退、重放、视频跳转**
  - **丢包和冗余**

流式存储多媒体



客户端缓存及播放时延：补偿网络增加的时延和时延抖动

客户端缓存，播放

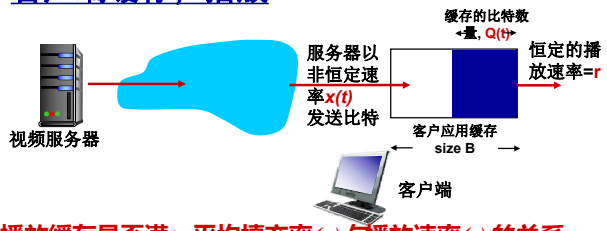


1、在时刻t=0，客户应用缓存为空，视频开始到达应用缓存，则建立Q比特的缓存所需要的时间是t_p

2、t_p时刻开始播放

3、Q的值随时间变化，取决于r和$x(t)$的值

客户端缓存，播放



播放缓存是否满：平均填充率(x)与播放速率(r)的关系

- $x < r$: 缓冲最终清空（造成视频播放暂停直到缓冲再次充满）
- $x > r$: 缓冲不空，缓存以x-r的速率从Q增加到B，提供的初始播放延迟足以承受变化的服务器发送速率$X(t)$

流式存储多媒体

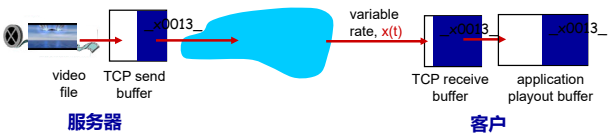
- 流式视频系统分为三种类型:
  - UDP流
  - HTTP流
  - 适应性HTTP流

流式多媒体：UDP

- 服务器以适配客户端播放速率的速率发送数据
  - 通常：发送速率=编码速率，且均为恒定速率
  - 传输速率达到可以忽略拥塞的级别
- 短的播放延迟（2-5秒）来消除网络时延抖动
- 使用RTP[RFC 2326]类型的多媒体载荷
- UDP可能无法通过防火墙

流式多媒体：HTTP

- 通过HTTP GET方法检索多媒体文件
- 以TCP拥塞控制和流控制允许的最大速率进行传输



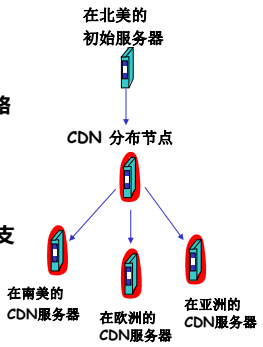
- 由于拥塞控制机制，服务器到客户的传输速率可能变化极大，分组也可能因为重发而导致延时
- 使用更大的播放时延来平滑TCP传输速率的变化。
- HTTP/TCP使得视频可以穿越防火墙和NAT

流式多媒体：DASH

- *DASH: Dynamic, Adaptive Streaming over HTTP*
- 允许用户使用不同的以太网接入速率流式播放具有不同编码速率的视频。
- 服务器:
  - 存放不同编码速率的视频版本。
  - 告示文件：为不同的视频版本提供URL及比特率
- 客户:
  - 咨询告示文件，获知不同版本
  - 定期测量服务器到客户端的带宽，根据当前带宽选择最大编码速率
  - 一次请求一块，对每块指定URL和要传输的字节范围
  - 可以在不同的时间点选择不同的编码率（取决于可用带宽的时间）

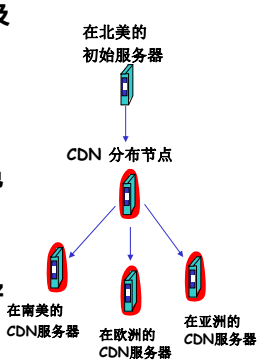
内容分发网络 (CDN)

- 从单一初始服务器（数据中心）实时地播放流式大文件(如视频)是一个挑战
  - 服务器到客户的分组可能需要跨越许多通信链路，并且可能通过许多ISP，这些通信链路提供的带宽可能小于视频消耗速率，导致时延较大。
  - 视频的多个副本经由相同的通信链路发送多次，浪费网络带宽且视频服务商需要向ISP 支付更多的费用
  - 存在dai单点故障问题



内容分发网络 (CDN)

- 解决方案: 将多媒体信息副本冗余在遍及因特网的数以百计的不同地理位置的服务器上（CDN）
  - 内容事先下载到CDN服务器上
  - 将内容放置在“靠近”用户处，避免跨越长路径发送内容损伤(丢包、时延)
  - 将每个用户的请求定向到能提供最好的用户体验的CDN
  - CDN服务器通常位于边缘/接入网络



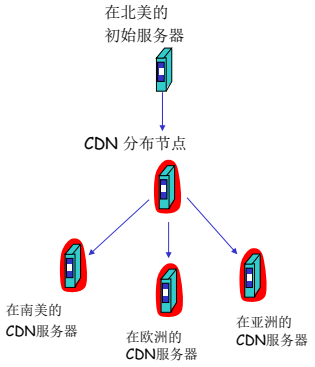
内容分发网络CDN

- 两种典型的CDN部署方法：
  - 推动CDN服务器深入到ISP的接入网，尽可能贴近用户：改善用户感受的时延和吞吐量，但维护和管理集群的任务成为挑战，例如Akamai
  - 延请做客：通过少量关键位置建造大集群，并使用专用高速网络连接这些集群，集群通常被放置在靠近许多第一层ISP的因特网交换节点，维护和管理费用较低，但付出用户较高时延和较低吞吐量的代价

内容分发网络 (CDN)

Akamai

- Akamai客户是内容提供商(如 CNN)
- 在Akamai的CDN服务器中复制客户的内容. 当提供商更新内容，CDN更新服务器



CDN集群选择策略

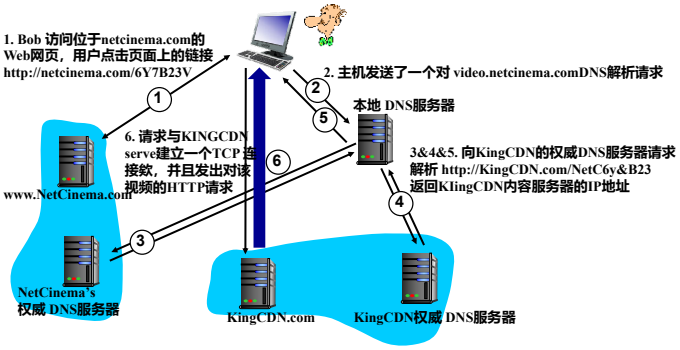
- CDN产生一幅“地图”，指示从叶ISP和CDN节点的距离
- 当请求到达权威DNS服务器：
  - 使用“地图”来决定最好的CDN服务器
  - 例如： CDN选择地理上“最接近”的集群

CDN访问实例

内容供应商NetCinema雇佣了第三方CDN公司KingCDN来向其客户分发视频gong

Bob (客户端) 请求视频 http://netcinema.com/6Y7B23V

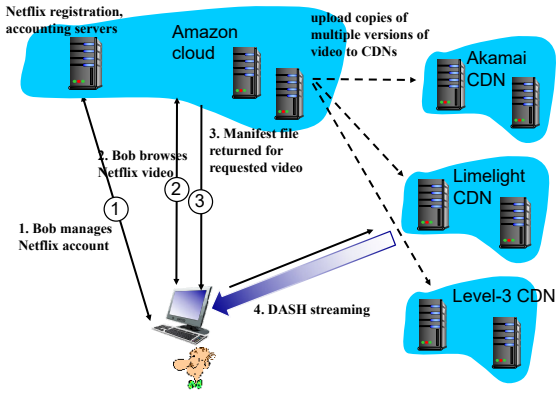
视频存储在CDN上的URL为 http://KingCDN.com/NetC6y&B23V



CDN案例学习: Netflix

- 2015年产生了美国约37%的下载量
- 两个主要部件：亚马逊云和自己的专用CDN
- 拥有自己的注册、计费服务、为用户提供电影浏览和搜索的应用系统
  - 运行于Amazon (第三方)云：
    - Netflix上传电影到Amazon云
    - 在云端创建多个版本的电影（不同的编码方式）
    - 从云端将不同版本上传到CDN
    - 云主机存储Netflix网页供用户浏览
    - 直接告诉用户使用特定的CDN服务器下载请求的电影资源

CDN案例学习: Netflix

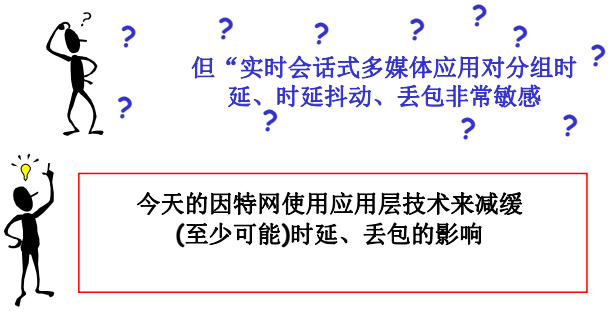


第五章 多媒体网络

- 多媒体网络应用
- 流式存储视频
- IP语音
- 实时会话式应用的协议
- 支持多媒体的网络

尽力而为服务的限制

TCP/UDP/IP: "尽力而为服务"对时延、丢包无 确保!



今天的因特网使用应用层技术来减缓(至少可能)时延、丢包的影响

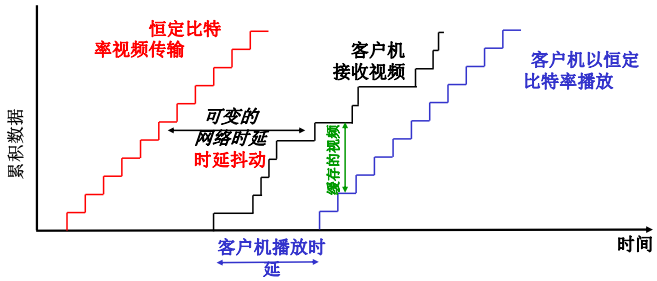
一个特定的VoIP案例

- 讲话者的语音：交互的语涌与静默期交替.
  - 仅在语涌期产生分组
  - 8 Kbytes/sec速率产生字节，每20ms将字节汇聚成块：160 字节数据
- 在每块上加上应用层首部
- 块+首部封装在UDP或TCP报文段中
- 在语涌期，应用程序每20ms向套接字发送报文段
- 对于"尽力而为"的因特网服务，接收方必须判断：
  - 什么时候播放一个块
  - 如何处理一个丢失块

IP语音:丢包、时延

- 网络丢包: 由于网络拥塞的IP数据报丢失 (路由器缓存溢出)
- 时延丢包: 在接收方，IP数据报到达太迟而无法播放
  - 端到端时延: 路由器中传输、处理、排队时延+链路中的传播时延+端系统(发送方，接收方)的处理 时延
  - 典型的最大可容忍时延: 400 ms
- 丢包容忍: 取决于语音编码和接收方处理丢包的方式，丢包率在1% 和10%之间可以容忍

IP语音：时延抖动



- 考虑两个连续分组的端到端时延：大于或小于20 msec
- 接受方如何消除时延抖动?

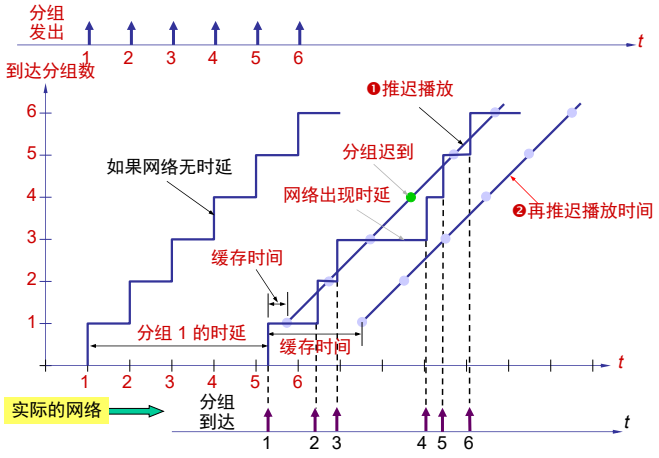
在接收方消除音频的时延抖动

- 在类似于因特网电话等实时应用中，接受方必须能够提供某种同步播放机制，使得在网络存在时延抖动的情况下，仍然能够正确地播放数字语音。
- 播放同步机制的实现：
  - 发送者对发送的每一数据块附加一个**序号**
  - 发送者对发送的每一数据块都附加数据块产生时间，即**时间戳**
  - 接收者对接收到的数据块**延迟**一段时间后，再予以**播放**。
    - 延迟时间必须保证绝大多数数据块都能在预期播放时间到达之前到达接收者。
    - 所有不能在预期播放时间之前到达的数据块被认为是丢失的数据块
    - 接收者可以忍受某种程度的数据块丢失
    - **固定延迟播放和自适应动态延迟播放**

IP语音: 固定播放时延

- 接收方在块生成后的q 毫秒来播放每个块
  - 块具有时间戳t: 在$t+q$播放块
  - 在$t+q$后块到达: 数据到达太迟而无法播放，数据"丢失"
- q的折衷选择:
  - 大q: 分组丢失少
  - 小q: 更好的交互体验

固定播放时延



自适应播放时延(1)

- **目的：** 最小化播放时延，更低的丢包率
- **方法：** 播放时延适应性调整:
  - 在每个语涌的开始时：估计网络时延以调整播放时延
  - 静默期压缩和伸长
  - 语涌期每20 msec仍播放块

在接收方平均时延的动态估计

$$d_i = (1-\alpha)d_{i-1} + \alpha\,(r_i - t_i)$$

- d_i: delay estimate after ith packet
- α: small constant, e.g. 0.1
- r_i: time received
- t_i: time sent (timestamp)
- $r_i - t_i$: measured delay of ith packet

自适应播放时延（2）

估计**时延的平均偏差**v_i也是有用的:

$$v_i = (1-b)v_{i-1} + b\,|r_i - t_i - d_i|$$

对收到的每一个分组计算d_i 和v_i 的估计值，但它们仅用于一个语涌的开始。
对语涌中的第一个分组，播放时间=$t_i + d_i + Kv_i$

其中K是一个正常数
语涌中的剩余分组定时地播放

自适应播放时延（3）

问题: 接收方怎样决定分组是否是一个语涌中的第一个分组?

- 如果无丢包，接收方看到连续的时戳
  - 连续时戳的差异> 20 msec -->语涌开始.
- 由于可能丢包，接收方必须看时戳和序号
  - 连续时戳的差值> 20 msec 和 没有间隙的序号--> 语涌 开始.

VoIP：丢包带来的麻烦

- 在诸如因特网电话等交互式实时应用中，重传丢失的分组通常是不合适的
  - 重传一个已经错过播放时间的分组毫无意义
  - 重传一个在路由器队列溢出的分组，通常无法在分组播放时间之前到达目标
- 需要讨论在保持可接收音频/视频质量的前提条件下，如何从丢包中恢复。

丢包恢复(1)

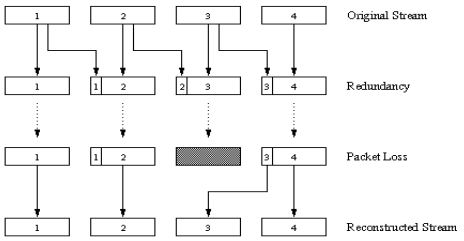
前向纠错(FEC): 简单的方案

- 对每组n个块，通过异或这n个初始块，生成一个冗余块
- 发送n+1块, 增加了1/n 的带宽
- 如果这n+1块至多丢失一个块，能够重构初始n块
- 播放时延需要固定为接收所有n+1分组的时间
- 折衷:
  - 增加n，浪费较少的带宽
  - 增加n，较长的播放时延
  - 增加n，2个或更多块丢失的概率增加

丢包恢复(2)

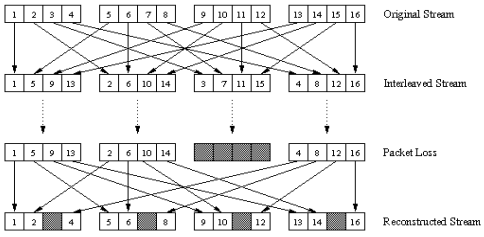
另一种FEC方案

- 发送一个较低分辨率的音频流作为冗余信息
- 例子： 64 kbps PCM 额定流和13 kbps GSM 冗余流



- 接收方能够隐藏任何时候的非连续丢包
- 也能够附加第 (n-1) 和 (n-2)低比特率块

丢包恢复(3)



交织: 块分成较小的单元

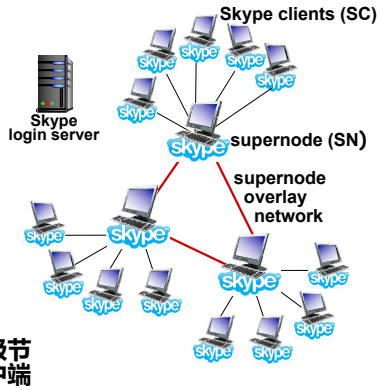
- 例子：每块4~5 msec 单元
- 分组包括来自不同块的小单元
- 如果分组丢失，仍有每个块的大部分
- 没有冗余开销
- 但增加了播放时延

丢包恢复(4)

- 在接收方对受损的音频流进行修复
- 产生一个类似原始替代数据来替代丢失的分组
  - “内插法”，用位于丢失分组之前和之后的音频，内插形成一个合适的分组来隐藏丢失
  - 重复：用位于丢失分组之前的，刚到达的分组的拷贝，来代替丢失的分组；对于较小的分组(4-40 ms)和低丢失率可表现出良好的性能

VoIP案例：Skype

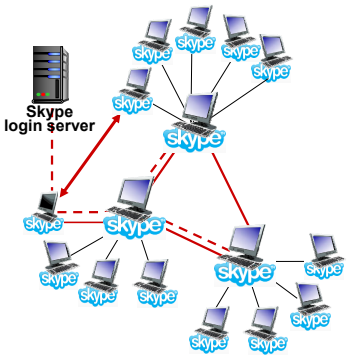
- 私有的应用层协议（通过逆向工程推导）
  - 所有控制消息及媒体信息加密
- P2P 组件：
  - 客户端(SC): Skype普通对等方，彼此直接互联并通信
  - 超级节点 (SN): 特殊功能的Skype超级对等方
  - 覆盖网络：部署于超级节点之间，用于定位客户端
  - 注册服务器（LS）



VoIP案例：Skype

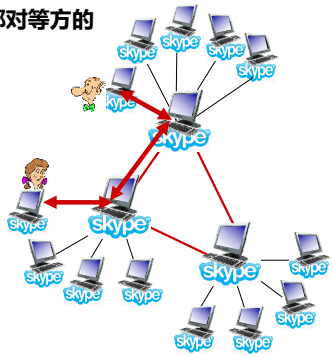
skype客户端操作:

1. 与SN联系，使用TCP加入Skype网络 (IP地址缓存)
2. 使用(用户名，口令)登录到集中式skype注册服务器
3. 从覆盖网络的超级节点，获取被叫的IP地址
   - 或客户端好友列表
4. 直接向被叫发起呼叫



Skype：点到点中继

- 问题：如果用户 Alice和Bob都在NAT之后?
  - NAT阻止外部对等方发起到内部对等方的连接
- 解决方案：使用中继和超级对等方
  - Alice和Bob保持与各自超级对等方SN的开放连接
  - Alice告知她的SN连接Bob
  - Alice的SN连接Bob的SN
  - Bob的SN利用Bob之前建立的，与SN的初始开放连接，与Bob通信



第五章 多媒体网络

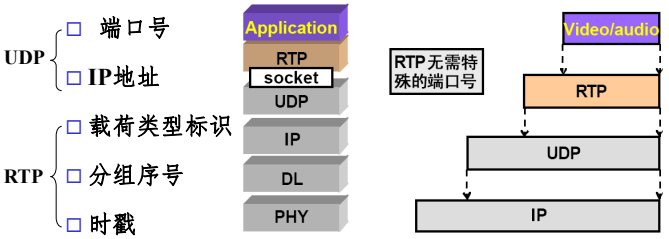
- 多媒体网络应用
- 流式存储视频
- IP语音
- 实时会话式应用的协议
- 支持多媒体的网络

实时传输协议(RTP)

- RTP定义了承载音频和视频数据的分组结构
- RFC 3550.
- RTP分组提供了
  - 载荷类型标识
  - 分组序号
  - 时戳
- RTP运行在端系统上
- RTP分组封装在UDP段中
- 交互能力: 如果两个因特网电话应用程序都集成了 RTP协议， 则它们能够相互通信

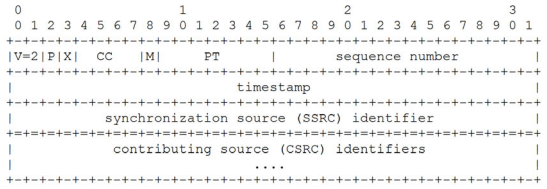
RTP运行在UDP之上

- RTP库或Java类，提供了扩展UDP的运输层接口：



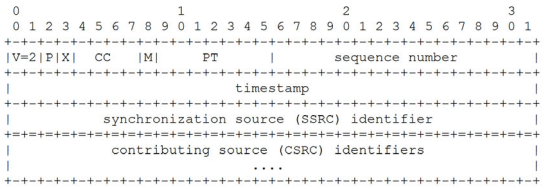
- RTP分组由首部和净荷两部分组成，净荷通常为数字化的语音数据，通过采样语音信号、量化采样信号、并对量化后的采样数据进行编码得到。净荷加上RTP首部后，以UDP数据报的形式发送。

RTP分组的封装



- 考虑经RTP发送64 kbps PCM编码语音
- 应用程序在块中收集编码数据，如每20 msec = 一个块中的160 字节
- 该音频块连同RTP首部形成了RTP分组，它被封装在UDP段中
- RTP首部指示了在每个分组中的音频编码类型
  - 发送方能够在一个会议中改变编码方式
- RTP首部也包含序号和时戳

RTP分组的封装



- 版本V：定义了RTP 的版本，值为2
- 填充P：若被设置，指明该RTP分组包含1个或多个附加在末端的填充比特。通常用于固定长度的加密算法或者底层数据单元中传输多个RTP 包
- 扩展X:若设置，则固定首部后面将跟随一个首部扩展
- CSRC计数（CC):包含了CSRC标识符的数目
- 标志M：用来允许在比特流中标记重要的事件，如帧边界

RTP首部(1)

RTP Header

- 载荷类型(7 bits): 指出当前正被使用的编码类型。如果发送方在会议中间改变编码，发送方通过该字段通知接收方
  - 载荷类型0: PCM mu-law, 64 kbps
  - 载荷类型3, GSM, 13 kbps
  - 载荷类型7, LPC, 2.4 kbps
  - 载荷类型26, Motion JPEG
  - 载荷类型31. H.261
  - 载荷类型33, MPEG2 video
- 序号 (16 bits): 对每个发送的RTP分组，序号自动增加，能够用于检测丢包和恢复分组顺序，序号初始值随机

RTP 首部(2)

RTP Header

- 时间戳字段 (32 比特长):反映RTP数据分组中的第一个字节的采样时间。如果RTP包是周期性产生的，那么将使用由采样时钟决定的采样时刻，而不是读取系统时间。
  - 例如：对于固定速率的音频，时间戳每个采样周期增加1 (例如，对8 KHz采样时钟，每125 μs为一个周期)
  - 如果一个音频应用从输入设备中读取含有160个采样周期的样本块，那么对每个块，时间戳的值增加160。
  - 时间戳可以用来实现不同媒体流的同步。

RTP 首部(2)

RTP Header

- SSRC字段 (32 比特长)：同步源标识符，标识RTP流的源。在RTP会话中的每个流应当具有一个独特的SSRC.
- CSRC列表：0~15项，识别在此RTP包中负载的所有贡献源。

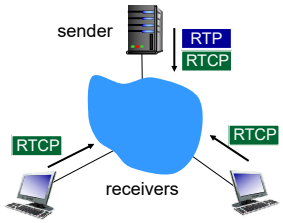
RTP和QoS

- RTP不提供确保数据定时交付的任何机制或提供服务质量保证
- RTP封装仅在端系统可见：而不被中间路由器所见
  - 提供尽力而为服务的路由器并不做任何特殊努力，以确保RTP分组以定时的方式到达目的地

实时控制协议(RTCP)

- 与RTP协同工作
- 在RTP会话中每个参与者周期性的向所有其他参与者传输RTCP控制分组
- 每个RTCP分组包含发送方和/或接收方报告
  - 报告对应用程序有用的统计参数：发送的分组数量，丢失的分组数量、到达时间的时延抖动等
- 反馈的信息能被用于控制性能
  - 发送方基于反馈信息可能修正它的传输

RTCP (续)



- 对于一个RTP会话，通常有单一的多播地址；属于该会话的所有RTP和RTCP分组使用该多播地址
- RTP和RTCP分组通过使用独特的端口号，RTCP端口号 =RTP数据流端口号+1
- 为了限制流量，当会议参与者增加时，每个参与者减小它的RTCP流量

RTCP分组类型

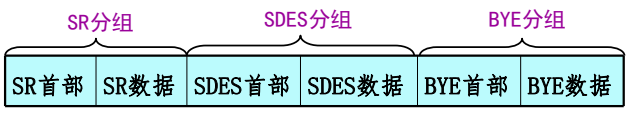
- RTCP定义了五种RTCP分组来携带不同类型的控制信息
  - 接收方报告分组:
    - 给出了该成员最近接收到的多媒体流的状况：丢包的分数，最后的序号，平均的到达间的时延抖动
  - 发送方报告分组:
    - 给出会话成员最近发送的多媒体流的状况：RTP流的SSRC, 当前的时间，发送分组的数量，和发送字节的数量

RTCP分组类型（续）

- RTCP定义了五种RTCP分组来携带不同类型的控制信息
  - 源描述分组:
    - 有关成员的描述信息：发送方的e-mail 地址, 发送方的名字，相联系RTP流的SSRC
    - 提供SSRC和用户/主机名字之间的映射
  - BYE分组
    - 成员离开会话时采用
  - APP分组
    - 用于携带和应用相关的信息，具体格式和含义在Profile中定义

RTCP复合分组

- 虽然上述RTCP分组类型是单独定义的，但它们并非逐个传输的。实际上，常常是两个或者多个RTCP分组被组合成单个复合分组后一起传输。
- RTCP协议规定，每个RTCP复合分组必须包含SR/RR和SDES分组，且必须以报告分组SR/RR开始。
- 当某个媒体数据流发送源需要发送SDES分组时，即使没有相关信息需要报告,也必须携带一个报告分组同时发送。因此，复合分组允许包含一个空的RR分组。



计算RTCP分组的发送周期

- RFC1889 提供了一种计算RTCP分组发送间隔的算法，保证了RTCP能适用于各种会话规模。
- 算法主要特征如下:
  - RTCP根据当前RTP会话中参与者个数，动态调整发送RTCP消息的间隔
  - 将当前会话带宽的5%用于RTCP消息
    - RTCP负载的25%分配给全部发送端
    - 全部接收端占用剩余的75%
- 例如：当前会话带宽2Mbps/s,发送方获得发送RTCP流量的速率是25 kbps，如果接收方之间平等地共享RTCP带宽，那么 对R个接收方，每个接收方获得发送RTCP流量的速率是 75/R kbps;

计算RTCP分组的发送周期

- 发送者和接收者的RTCP分组传输周期计算公式为

$$T_{发送} = \frac{发送端个数}{0.25*0.05*会话带宽}(RTCP平均包长)$$

$$T_{接收} = \frac{接收端个数}{0.75*0.05*会话带宽}(RTCP平均包长)$$

  - 实际的RTCP分组发送间隔=MAX(5,计算出的RTCP分组发送间隔)
  - 当RTCP复合分组长度增大时，必须降低发送速率，以限制RTCP分组占用的会话带宽比例。

回顾：流式存储音频和视频

- 应用级流式技术以最大限度利用“尽力而为”服务:
  - Client请求存储在某服务器（通常是WEB服务器)上的已压缩的音频/视频文件
  - 被请求的音频/视频文件被封装成RTP分组
  - RTP分组传输给UDP套接字，将其封装在UDP报文段中（不用TCP)
  - Client一般在发出请求几秒后收到流就开始播放
  - Client若需要暂停/恢复、跳转等交互式功能则通过RTSP协议实现

流式媒体的用户控制: RTSP

HTTP

- 不能针对多媒体内容
- 没有用于快进的命令等

RTSP: RFC 2326

- 客户机-服务器应用层协议
- 为用户控制播放：倒带，快进，暂停，恢复，重定位等...

RTSP无法胜任的工作:

- 不能定义音频/视频怎样为经网络传输的流式而封装
- 不能约定流式媒体如何传输；它能够经UDP或TCP传输
- 不能定义媒体播放器怎样缓存音频/视频

RTSP: 带外控制

FTP使用一个 “带外”控制信道:

- 文件传输通过一条TCP连接
- 控制信息(目录变化、文件删除、文件更名等)经一条单独的TCP连接发送
- “带外”和“带内”信道使用不同的端口号

RTSP报文也在带外发送:

- RTSP控制报文使用与媒体流不同的端口号：带外
  - 端口554
- 媒体流被认为是“带内”

多媒体会话

- RTSP可能会同时控制一个或者多个媒体流，这些媒体流被称为多媒体会话。
  - 一个多媒体会话采用标准的URL格式来标示。
    - 例：rtsp://media.example.com:544/twister
      - 标识了media.example.com上的一个名为twister的会话。
  - 会话的相关信息通过一个相应的展现描述文件来定义，包括：有哪些媒体流，每个媒体流的编码方式、传输地址和媒体流携带的内容的有关描述等。

多媒体会话

- RTSP支持任何会话描述语言，包括IETF定义的SDP协议和W3C的SMIL等。
  - 假设某个会话包含了一个音频流和一个视频流，会话描述采用SMIL语言：展现描述文件实例
  - 通过rtsp://media.example.com:544/twister 可以同时控制音频/视频流。
  - 如果只需要控制会话的一部分，如音频流，则可以通过：rtsp://media.example.com:544/twister/audiotrack

元文件（展现描述文件）实例之一

```
<title>Twister<title>
    <session>
      <group language=en lipsync>
        <track type=audio e="DVI4/16000/2" pt="90 DVI4/8000/1"
          src="rtsp://media.example.com:544/twister/audiotrack">
        <track type="video/jpeg"
          src="rtsp://media.example.com:544/twister/videotrack">
      <group>
    </session>
```

用于描述一个会话中多个媒体流的信息

元文件（展现描述文件）实例之二

```
<title>Twister</title>
<session>
    <group language=en lipsync>
        <switch>
            <track type=audio
                e="PCMU/8000/1"
                src = "rtsp://audio. exmaple.com/twister/audio.en/lofi">
            <track type=audio
                e="DVI4/16000/2" pt="90 DVI4/8000/1"
                src="rtsp://audio. exmaple.com/twister/audio.en/hifi">
        </switch>
        <track type="video/jpeg"
                src="rtsp://video.exmaple.com/twister/video">
    </group>
</session>
```

指向几个连续媒体文件的引用

RTSP消息

- RTSP消息的传输
  - RTSP消息采用TCP或UDP来传输，且消息的传输与RTP协议独立；如果采用TCP，则URL的协议字段为rtsp，而如果采用UDP，则为rtspu。
  - RTSP提供了相应的机制来保证RTSP消息传输的可靠。
- RTSP消息的组成：
  - RTSP请求消息(request)
  - RTSP回应消息(response)。

RTSP消息（RFC2326）

- 请求消息(request)

Method	SP	Request-URI	SP	RTSP-Version	CR	LF
Message Header					CR	LF
Message Body					CR	LF

RTSP-Version	Status-Code	SP	Reason-Phrase	CR	LF
Message Header				CR LF	CR LF
Message Body					CR LF

其中 Method：请求命令； SP ：空白符； Request-URI ：媒体表示地址；CR：回车； LF：换行； RTSP-Version：协议版本号； Message Header：消息头；Message Body ：消息体； Status-Code ：3 位状态码，用于回应请求时表示主机状态 ；Reason-Phrase：是与状态码对应的文本解释。

RTSP Message Header选项

消息头/报头	类型	协议定义	消息头/报头	类型	协议定义
Accept	Req	Optional	Public	Res	Optional
Connection	G	Required	Range	Res	Optional
Content-Encoding	E	Required	Require	Req	Required
Content-Language	E	Required	RTP-Info	Res	Required
Content-Length	E	Required	Session	Req/Res	Required
Content-Location	E	Optional	Transport	Req/Res	Required
Content-Type	Res	Required	Unsupported	Req	Required
Proxy-Required	Req	Required	User-Agent	Req	Optional
CSeq	G	Required			

类型中“G”表示应用于请求和响应消息；“Req”表示应用于请求消息；“Res”表示应用于响应消息；“E”表示应用于实体头字段。

一个简单的rtsp交互过程

- Step1:

  C->S:OPTION request //询问S有哪些Method(方法)可用

  S->C:OPTION response //S回应信息中包括所有可用方法

- Step2:

  C->S:DESCRIBE request //要求得到S提供的会话描述信息

  S->C:DESCRIBE response //S回应会话描述信息，可采用IETF的SDP协议描述

  *在获得媒体流的信息之后，客户端启动相应的媒体应用程序来准备处理媒体流。

C表示rtsp客户端，S表示rtsp服务端

RTSP 方法列表

方法	方向	对象	要求	含义
DESCRIBE	C->S	P, S	推荐	检查演示或媒体对象的描述，也允许使用接收头指定用户理解的描述格式。ESCRIBE的答复-响应组成媒体RTSP初始阶段
ANNOUNCE	C->S S->C	P, S	可选	当从用户发往服务器时，ANNOUNCE将请求URL识别的演示或媒体对象描述发送给服务器；反之，ANNOUNCE实时更新连接描述。如新媒体流加入演示，整个演示描述再次发送，而不仅仅是附加组件，使组件能被删除
GET_PARAMETER	C->S S->C	P, S	可选	GET_PARAMETER请求检查RUL指定的演示与媒体的参数值。没有实体体时，GET_PARAMETER也许能用来测试用户与服务器的连通情况
OPTIONS	C->S S->C	P, S	要求	可在任意时刻发出OPTIONS请求，如用户打算尝试非标准请求，并不影响服务器状态

RTSP方法列表

方法	方向	对象	要求	含义
PAUSE	C->S	P, S	推荐	PAUSE请求引起流发送临时中断。如请求URL命名一个流，仅回放和记录被停止；如请求URL命名一个演示或流组，演示或组中所有当前活动的流发送都停止。恢复回放或记录后，必须维持同步。在SETUP消息中连接头超时参数所指定时段期间被暂停后，尽管服务器可能关闭连接并释放资源，但服务器资源会被预订
PLAY	C->S	P, S	要求	PLAY告诉服务器以SEFUP指定的机制开始发送数据；直到一些SETUP请求被成功响应，客户端才可发布PLAY请求。PLAY请求将正常播放时间设置在所指定范围的起始处，发送流数据直到范围的结束处。PLAY请求可排成队列，服务器将PLAY请求排成队列，顺序执行

RTSP方法列表

方法	方向	对象	要求	含义
RECORD	C->S	P, S	可选	该方法根据演示描述初始化媒体数据记录范围，时标反映开始和结束时间；如没有给出时间范围，使用演示描述提供的开始和结束时间。如连接已经启动，立即开始记录，服务器数据请求URL或其他URL决定是否存储记录的数据；如服务器没有使用URL请求，响应应为201(创建)，并包含描述请求状态和参考新资源的实体与位置头。支持现场演示记录的媒体服务器必须支持时钟范围格式，smpte格式没有意义
REDIRECT	S->C	P, S	可选	重定向请求通知客户端连接到另一服务器地址。它包含强制头地址，指示客户端发布URL请求；也可能包括参数范围，以指明重定向何时生效。若客户端要继续发送或接收URL媒体，客户端必须对当前连接发送TEARDOWN请求，而对指定主执新连接发送SETUP请求

RTSP方法列表

方法	方向	对象	要求	含义
SETUP	C->S	S	要求	对URL的SETUP请求指定用于流媒体的传输机制。客户端对正播放的流发布一个SETUP请求，以改变服务器允许的传输参数。如不允许这样做，响应错误为"455 Method Not Valid In This State"。为了透过防火墙，客户端必须指明传输参数，即使对这些参数没有影响
SET_PARAMETER	C->S S->C	P, S	可选	这个方法请求设置演示或URL指定流的参数值。请求仅应包含单个参数，允许客户端决定某个特殊请求为何失败。如请求包含多个参数，所有参数可成功设置，服务器必须只对该请求起作用。服务器必须允许参数可重复设置成同一值，但不让改变参数值。注意：媒体流传输参数必须用SETUP命令设置。将设置传输参数限制为SETUP有利于防火墙。将参数划分成规则排列形式，结果有更多有意义的错误指示

RTSP方法列表

方法	方向	对象	要求	含义
TEARDOWN	C->S	P, S	要求	TEARDOWN请求停止给定URL流发送，释放相关资源。如URL是此演示URL，任何RTSP连接标识不再有效。除非全部传输参数是连接描述定义的，SETUP请求必须在连接可再次播放前发布

一个简单的RTSP交互过程

- **Step3:**

C->S:SETUP request //设置会话的属性、传输模式，请求S建立会话

S->C:SETUP response //S建立会话、返回会话标识符以及会话相关信息

***同时服务方将分配相应的资源，比如媒体流将要传输的网络套接字，预约网络带宽等。**

一个简单的RTSP交互过程

- Step4:

  C->S:PLAY request　//C请求播放头部中的URL指定的媒体流，同时客户端还指定了播放的起始时间和结束时间

  S->C:PLAY response //S回应该请求的信息

  S->C: 持续发送流媒体数据

- Step5:

  C->S:TEARDOWN request　//C请求关闭会话

  S->C:TEARDOWN response　//S回应该请求

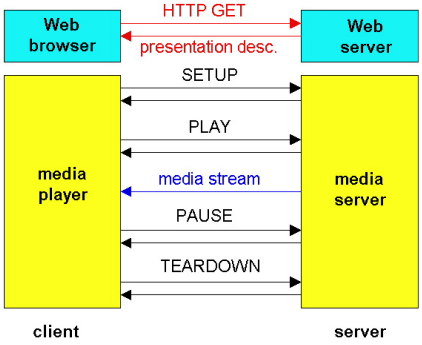
  *拆除会话，释放分配的资源。

一个简单的RTSP交互过程

- 上述的过程是标准的RTSP流程，在实际应用中：
  - Step3和Step4是必需的。
  - 对于Step1，只要服务器客户端约定好，有哪些方法可用，则option请求可以省略。
  - 对于Step2，如果有其他途径得到媒体初始化描述信息(例如通过http请求等），则不需要使用describe请求。
  - 对于Step5，可以根据系统需求的设计来决定是否需要。
  - 还可以根据需要，增加PAUSE方法暂停媒体流的传输

RTSP交互的实例

- 服务器将元文件传送给Web浏览器
- 浏览器调用播放器
- 播放器向流式服务器建立一条控制连接和一条数据连接



RTSP交互的实例之一

C->M: DESCRIBE rtsp://twister RTSP/1.0
CSeq: 1

M->C: RTSP/1.0 200 OK
CSeq: 1
Content-Type: application/sdp
v=0
o=-2890844526 2890842807 IN IP4 192.16.24.202
s=RTSP Session
m=audio 0 RTP/AVP 0
a=control:rtsp://audio.example.com/twister/audio.en
m=video 0 RTP/AVP 31
a=control:rtsp://video.example.com/twister/video

Step1：为了得到Server上关于twister会话的描述信息(SDP描述)，Client向Server发起DESCRIBE请求

SDP的格式

v=（协议版本）
o=（所有者/创建者和会话标识符）
s=（会话名称）
i=*（会话信息）
u=*（URI 描述）
e=*（Email 地址）
p=*（电话号码）
c=*（连接信息）
b=*（带宽信息）
z=*（时间区域调整）
k=*（加密密钥）
a=*（0 个或多个会话属性行）

时间描述：
t=（会话活动时间）
r=*（0或多次重复次数）
媒体描述：
m=(媒体名称和传输地址)
i=*（媒体标题）
c=*（连接信息-如果包含在会话层则该字段可选）
b=*（带宽信息）
k=*（加密密钥）
a=*（0 个或多个媒体属性行）

RTSP交互的实例之一

C->A: SETUP rtsp://audio.example.com/twister/audio.en RTSP/1.0
CSeq: 1
Transport: RTP/AVP/UDP;unicast;client_port=3056-3057

A->C: RTSP/1.0 200 OK
CSeq: 1
Session: 12345678
Transport: RTP/AVP/UDP;unicast;client_port=3056-3057;
server_port=5000-5001

C->V: SETUP rtsp://video.example.com/twister/video RTSP/1.0
CSeq: 1
Transport: RTP/AVP/UDP;unicast;client_port=3058-3059

V->C: RTSP/1.0 200 OK
CSeq: 1
Session: 23456789
Transport: RTP/AVP/UDP;unicast;client_port=3058-3059;
server_port=5002-5003

Step2：Client发起SETUP请求，分别和相应的媒体服务器建立RTSP会话；服务器响应返回会话标识

RTSP交互的实例之一（2）

```
C -> V:  PLAY rtsp://video.example.com/twister/video RTSP/1.0
         CSeq: 2
         Session: 23456789
         Range: smpte=0:10:00 -

V -> C:  RTSP/1.0 200 OK
         CSeq: 2
         Session: 23456789
         Range: smpte=0:10:00 - 0:20:00
         RTP-Info: url=rtsp://video.example.com/twister/video;
                   seq=12312232;rtptime=78712811
```

Step3：Client发送PLAY请求进行播放，SMTPE时间戳指定了从媒体流的第10分钟以后开始播放直到结束

```
C -> A:  PLAY rtsp://audio.example.com/twister/audio.en RTSP/1.0
         CSeq: 2
         Session: 12345678
         Range: smpte=0:10:00 -
A -> C:  RTSP/1.0 200 OK
         CSeq: 2
         Session: 12345678
         Range: smpte=0:10:00 - 0:20:00
         RTP-Info:url=rtsp://audio.example.com/twister/audio.en;
                  seq=876655;rtptime=1032181
```

RTSP交互的实例之一（3）

```
C -> A:  TEARDOWN rtsp://audio.example.com/twister/audio.en RTSP/1.0
         CSeq: 3
         Session: 12345678

A -> C:  RTSP/1.0 200 OK
         CSeq: 3
```

Step4： Client发送TEARDOWN请求结束会话，Server释放前面分配的资源。

```
C -> V:  TEARDOWN rtsp://video.example.com/twister/video RTSP/1.0
         CSeq: 3
         Session: 23456789

V → C:  RTSP/1.0 200 OK
         CSeq:3
```

元文件例子

```
<title>Twister</title>
<session>
    <group language=en lipsync>
        <switch>
            <track type=audio
                e="PCMU/8000/1"
                src = "rtsp://audio.exmaple.com/twister/audio.en/lofi">
            <track type=audio
                e="DVI4/16000/2" pt="90 DVI4/8000/1"
                src="rtsp://audio.exmaple.com/twister/audio.en/hifi">
        </switch>
        <track type="video/jpeg"
                src="rtsp://video.exmaple.com/twister/video">
    </group>
</session>
```

RTSP交换例子

```
C: SETUP rtsp://audio.exmaple.com/twister/audio RTSP/1.0
   Transport: rtp/udp; compression; port=3056; mode=PLAY

S: RTSP/1.0 200 1 OK
   Session 4231

C: PLAY rtsp://audio.exmaple.com/twister/audio.en/lofi RTSP/1.0
   Session: 4231
   Range: npt=0-

C: PAUSE rtsp://audio.exmaple.com/twister/audio.en/lofi RTSP/1.0
   Session: 4231
   Range: npt=37

C: TEARDOWN rtsp://audio.exmaple.com/twister/audio.en/lofi RTSP/1.0
   Session: 4231

S: 200 3 OK
```

SIP：会话发起协议(Session Initiation Protocol)

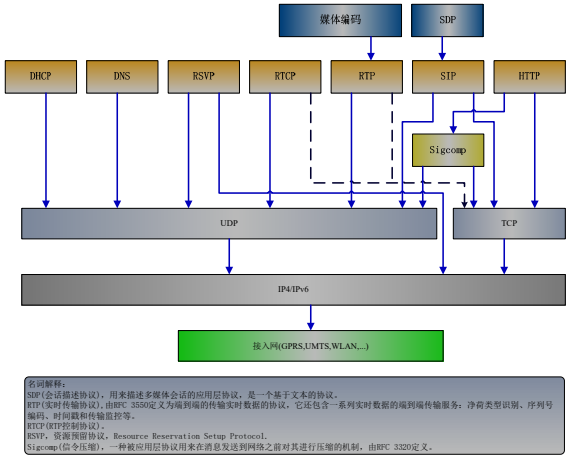
SIP展望

- 所有电话呼叫和视频会议呼叫在因特网上发生
- 人们用名字或电子邮件地址标识，而不是用电话号码
- 你能够找到被被叫方，无论他漫游到哪里，无论他当前使用了何种IP设备

SIP：最热门且成熟的通信协议

- 通信提供商及其合作伙伴和用户越来越渴求新一代基于 IP 的服务。SIP 是第一个适合各种媒体内容而实现多用户会话的协议，现在已成了 Internet 工程任务组 (IETF) 的规范。
- SIP 规定了以下基本的通信要求：
  - 用户定位服务
  - 会话建立
  - 会话参与方管理
  - 特点的有限确定
- SIP 的重要特点：它不定义要建立的会话的类型，而只定义应该如何管理会话。
- 这种灵活性使 SIP 可以用于众多应用和服务中，包括交互式游戏、音乐和视频点播以及语音、视频和 Web 会议。

SIP在协议栈中的位置



SIP消息

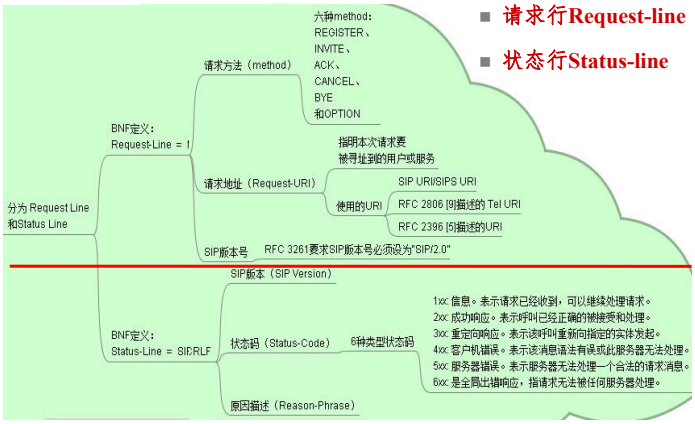
- SIP消息是SIP客户端和服务器之间通信的基本信息单元
- SIP消息基于文本,采用UTF-8编码中的ISO 10646字符集
- SIP协议借鉴了HTTP协议的设计思想,很多消息格式与之相同
- SIP协议支持UDP传输协议

SIP消息结构

- SIP消息由：
  - 起始行Start-line
  - 一个或多个字段组成的消息头Message-header
  - 空行CRLF
  - 消息体Message body（可选项）

起始行

- 起始行分为：
  - 请求行Request-line
  - 状态行Status-line



SIP消息分类

- SIP消息分为:请求消息和应答消息
- 当服务器收到一个SIP请求消息，并对其进行解析后，根据不同的请求，要返回一个或多个SIP应答消息

应答消息的6种状态码

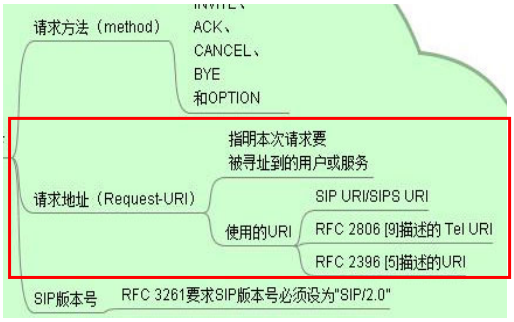
1xx: 信息。表示请求已经收到，可以继续处理请求。
2xx: 成功响应。表示呼叫已经正确的被接受和处理。
3xx: 重定向响应。表示该呼叫重新向指定的实体发起。
4xx: 客户机错误。表示该消息语法有误或此服务器无法处理。
5xx: 服务器错误。表示服务器无法处理一个合法的请求消息。
6xx: 是全局出错响应，指请求无法被任何服务器处理。

SIP请求消息的6种方法

- INVITE：邀请用户参加一个会话
- ACK：用于客户机向服务器证实自己收到对INVITE请求的最终响应
- OPTION：用于向服务器查询其能力
- BYE：呼叫的一方在释放呼叫之前发出该请求，收到请求的一方必须停止向被请求方法送媒体流
- CANCEL：用于取消一个CALL-ID
- REGISTER：用于客户机向SIP服务器注册列在TO字段中的地址信息

SIP URI

- SIP协议对位于某个主机的用户，使用SIP通用资源定位器URI进行标识，并根据该URI进行寻址



SIP URI

- 格式：sip:用户信息@主机端口|参数||消息头|
  用户信息：用户名或电话号码
  主机端口：域名或数字形式的网络地址和端口
  参数：定义具体的URI参数，例如传输协议、生存时间等
  消息头：用来传递额外信息

SIP URI

- 两种SIP URI：
  - Address-of-Record(AOR): 用于标示一个用户，例如sip:bob@nokia.com(需要DNS SRV记录来定位nokia.com域的SIP服务器)
  - 主机的FQDN或IP地址：例如，sip:bob@211.136.85.58，或sip:bob@workstation1.nokia.com(不需要路由解析)

Tel URI

- 格式：一个全球号码或者本地号码
  - 全球号码：遵从E.164号码规则，以"+"开始
  - 本地号码：需要有电话上下文(phone-context)参数来标识号码范围
- 例如：
  - 全球号码：tel:+358-9-123-45678
  - 具有域名上下文的本地号码：tel:45678;phone-context=example.com
  - 具有全球号码上下文的本地号码：tel:45678;phone-context=+358-9-123

消息头

- 通用头general-header
- 响应头respone-header
- 请求头request-header
- 实体头entity-header

4类消息头的作用

- 通用头general-header：描述消息基本属性，可用于请求消息和应答消息
- 请求头request-header：只能用于请求消息，被用来传递有关请求的附加信息，对请求消息进行补充说明
- 响应头respone-header：只能用于应答消息，用来传递有关应答的附加信息，对应答消息进行补充说明
- 实体头entity-header：用于描述消息体的长度，格式，编码类型等属性，可用于请求消息和应答消息

消息头格式说明

- 每个消息头都是一个“句子”，格式遵循RFC 822 Internet文本消息格式标准
- 由字段名和值域两部分组成，中间以“：”分隔，值域的内容与具体消息头相关

SIP消息实例 – INVITE消息

INVITE sip:victor@add.ultrapower.com.cn;transport=tcp SIP/2.0
Via: SIP/2.0/TCP 192.168.5.162;branch=z9hG4bK381ACAFC.537D4376;branched=FALSE;
Max-Forwards: 69
Contact: <sip:long@add.ultrapower.com.cn:2992;maddr=192.168.5.162;transport=tcp;ms-received-cid=B100>
Via: SIP/2.0/TCP 192.168.5.162:11499;ms-received-port=2992;ms-received-cid=b100
Record-Route: <sip:helen.add.ultrapower.com.cn;transport=tcp>;tag=9B8AE5F9C759FB02F679AFD6CB71394C
From: <sip:long@add.ultrapower.com.cn>;tag=704b1683edd4438f85a34adb6201e078;epid=d9176cec0c
To: <sip:victor@add.ultrapower.com.cn>;epid=203e516cd9
Call-ID: d455d3faf197403482154b9a319f54e0
CSeq: 1 INVITE
Require: com.microsoft.rtc-multiparty
Content-Type: application/sdp
Content-Length: 134

SIP消息实例 – INVITE的起始行

INVITE sip:victor@add.ultrapower.com.cn;transport=tcp SIP/2.0
Via: SIP/2.0/TCP 192.168.5.162;branch=z9hG4bK381ACAFC.537D4376;branched=FALSE;
Max-Forwards: 69
Contact: <sip:long@add.ultrapower.com.cn:2992;maddr=192.168.5.162;transport=tcp;ms-received-cid=B100>
Via: SIP/2.0/TCP 192.168.5.162:11499;ms-received-port=2992;ms-received-cid=b100
Record-Route: <sip:helen.add.ultrapower.com.cn;transport=tcp>;tag=9B8AE5F9C759FB02F679AFD6CB71394C
From: <sip:long@add.ultrapower.com.cn>;tag=704b1683edd4438f85a34adb6201e078;epid=d9176cec0c
To: <sip:victor@add.ultrapower.com.cn>;epid=203e516cd9

- SIP消息的第一部分：起始行(StartLine)

  本消息属于Request Line，它所包含的信息：

  RequestType: 方法名就是INVITE，表示邀请其他用户加入会话

  Request-URI: sip:long@add.ultrapower.com.cn;transport=tcp, 这属于SIP URI中的“Address-of-Record(AOR)”种类。

  SIP Version:2.0版本号

SIP消息实例 – INVITE的消息头

INVITE sip:victor@add.ultrapower.com.cn;transport=tcp SIP/2.0
Via: SIP/2.0/TCP 192.168.5.162;branch=z9hG4bK381ACAFC.537D4376;branched=FALSE;
Max-Forwards: 69
Contact: <sip:long@add.ultrapower.com.cn:2992;maddr=192.168.5.162;transport=tcp;ms-received-cid=B100>
Via: SIP/2.0/TCP 192.168.5.162:11499;ms-received-port=2992;ms-received-cid=b100
Record-Route: <sip:helen.add.ultrapower.com.cn;transport=tcp>;tag=9B8AE5F9C759FB02F679AFD6CB71394C

- SIP消息的第二部分：消息头(Message Header)

  Via：记录了请求在SIP网络中的路由路径。

  最下面的Via是初始化这个请求的UA(User Agent)插入的；

  上面的Via都是在这个路由路径上的Proxy们插入的。

Via头域就是用来指示如何将响应沿原路返回到UA的。

SIP消息实例 – INVITE的消息头

INVITE sip:victor@add.ultrapower.com.cn;transport=tcp SIP/2.0
Via: SIP/2.0/TCP 192.168.5.162;branch=z9hG4bK381ACAFC.537D4376;branched=FALSE;
Max-Forwards: 69
Contact: <sip:long@add.ultrapower.com.cn:2992;maddr=192.168.5.162;transport=tcp;ms-received-cid=B100>
Via: SIP/2.0/TCP 192.168.5.162:11499;ms-received-port=2992;ms-received-cid=b100
Record-Route: <sip:helen.add.ultrapower.com.cn;transport=tcp>;tag=9B8AE5F9C759FB02F679AFD6CB71394C
From: <sip:long@add.ultrapower.com.cn>;tag=704b1683edd4438f85a34adb6201e078;epid=d9176cec0c
To: <sip:victor@add.ultrapower.com.cn>;epid=203e516cd9

- SIP消息的第二部分：消息头(Message Header)

  Max-Forwards：最大转发数，用来限制一个SIP请求消息所能经过的实体的最大数目。

  如果消息拷贝中包含一个Max-Forwards字段，代理服务器必须将其减一。

  如果消息拷贝中不含Max-Forwards字段，代理服务器必须加入该字段，其值应该为70。

  如果请求消息中包含Max-Forwards字段，且其值为零，则SIP实体不得转发，并且返回483(Too Many Hops)响应。

SIP消息实例 – INVITE的消息头

- SIP消息的第二部分：消息头(Message Header)

  From Tag，To Tag，和Call-ID构成了dialog信息，可以唯一标识一个dialog。

  在本次呼叫(Call)中的所有请求和响应将使用同样dialog信息。

  Call-ID的构成：伪随机数字+”@”+主机名或者IP地址

Via: SIP/2.0/TCP 192.168.5.162:11499;ms-received-port=2992;ms-received-cid=b100
Record-Route: <sip:helen.add.ultrapower.com.cn;transport=tcp>;tag=9B8AE5F9C759FB02F679AFD6CB71394C
From: <sip:long@add.ultrapower.com.cn>;tag=704b1683edd4438f85a34adb6201e078;epid=d9176cec0c
To: <sip:victor@add.ultrapower.com.cn>;epid=203e516cd9
Call-ID: d455d3faf197403482154b9a319f54e0
CSeq: 1 INVITE
Require: com.microsoft.rtc-multiparty
Content-Type: application/sdp
Content-Length: 134

SIP消息实例 – INVITE的消息头

- SIP消息的第二部分：消息头(Message Header)

  CSeq: Command Sequence Number

  用于标识事务并对事务排序。由一个序列号和请求方法组成。

  呼叫开始时初始化CSeq。非REGISTER请求的CSeq，序列号值可以是一个任意的32位无符号整数。CSeq按照各自方向严格单向按1递增。

  CSeq能够区分某个请求是新请求还是重发的请求。

From:
<sip:long@add.ultrapower.com.cn>;tag=704b1683edd4438f85a34adb6201e078;epid=d9176cec0c
To: <sip:victor@add.ultrapower.com.cn>;epid=203e516cd9
Call-ID: d455d3faf197403482154b9a319f54e0
CSeq: 1 INVITE
Require: com.microsoft.rtc-multiparty
Content-Type: application/sdp
Content-Length: 134

SIP消息实例 – INVITE的消息头

INVITE sip:victor@add.ultrapower.com.cn;transport=tcp SIP/2.0
Via: SIP/2.0/TCP 192.168.5.162;branch=z9hG4bK381ACAFC.537D4376;branched=FALSE;
Max-Forwards: 69
Contact:
<sip:long@add.ultrapower.com.cn:2992;maddr=192.168.5.162;transport=tcp;ms-received-cid=B100>

- SIP消息的第二部分：消息头(Message Header)

  Contact: 包含的SIP/SIPS URI是UA希望用来接收请求的地址，后续请求可以用它来联系到当前UA。

  如果代理服务器没有插入Record-Route字段来希望自己留在后续请求消息的传输路径上，那么可以忽略这些代理服务器，后续请求直接用Contact字段的URI来通讯。

  当Contact中包含一个显示名称时，带有所有的URI参数的URI应该放入尖括号<>中。

SIP消息实例 – INVITE的消息头

INVITE sip:victor@add.ultrapower.com.cn;transport=tcp SIP/2.0
Via: SIP/2.0/TCP 192.168.5.162;branch=z9hG4bK381ACAFC.537D4376;branched=FALSE;
Max-Forwards: 69
Contact: <sip:long@add.ultrapower.com.cn:2992;maddr=192.168.5.162;transport=tcp;ms-received-cid=B100>
Via: SIP/2.0/TCP 192.168.5.162:11499;ms-received-port=2992;ms-received-cid=b100
Record-Route:
<sip:helen.add.ultrapower.com.cn;transport=tcp>;tag=9B8AE5F9C759FB02F679AFD6CB71394C
From:
<

- SIP消息的第二部分：消息头(Message Header)

  Record-Route:

  如果代理服务器希望自己留在后续请求消息的传输路径上，而这些后续请求消息属于由当前请求创建的一个对话，那么即使请求中已经出现了Route字段，代理服务器也必须在请求消息的拷贝中已有的Record-Route字段值之前插入一个Record-Route字段。

SIP消息实例 – INVITE的消息头

- SIP消息的第二部分：消息头(Message Header)

  Content-Type: 定义消息实体的类型，如text/plain，或application/sdp。

  如果消息体不为空，Content-Type字段必须存在。

  Content-Length: 定义消息实体的长度，单位为字节。

  SIP消息实体不能采用HTTP 1.1中所定义的“Chunked”传送编码机制。

Via: SIP/2.0/TCP 192.168.5.162:11499;ms-received-port=2992;ms-received-cid=b100
Record-Route:
<sip:helen.add.ultrapower.com.cn;transport=tcp>;tag=9B8AE5F9C759FB02F679AFD6CB71394C
From:
<sip:long@add.ultrapower.com.cn>;tag=704b1683edd4438f85a34adb6201e078;epid=d9176cec0c
To: <sip:victor@add.ultrapower.com.cn>;epid=203e516cd9
Call-ID: d455d3faf197403482154b9a319f54e0
CSeq: 1 INVITE
Require: com.microsoft.rtc-multiparty
Content-Type: application/sdp
Content-Length: 134

SIP消息实例 – INVITE的消息头

- SIP消息的第二部分：消息头(Message Header)

  Require:

  UAC通过Require字段列出的选项标签，告知UAS处理请求时需要支持的选项，本字段为可选，但不可以被忽略。

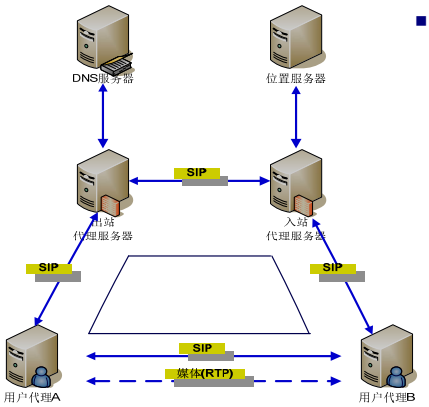
Via: SIP/2.0/TCP 192.168.5.162:11499;ms-received-port=2992;ms-received-cid=b100
Record-Route:
<sip:helen.add.ultrapower.com.cn;transport=tcp>;tag=9B8AE5F9C759FB02F679AFD6CB71394C
From:
<sip:long@add.ultrapower.com.cn>;tag=704b1683edd4438f85a34adb6201e078;epid=d9176cec0c
To: <sip:victor@add.ultrapower.com.cn>;epid=203e516cd9
Call-ID: d455d3faf197403482154b9a319f54e0
CSeq: 1 INVITE
Require: com.microsoft.rtc-multiparty
Content-Type: application/sdp
Content-Length: 134

SIP 消息实例– INVITE的消息体

v=0
o=-0 0 IN IP4 192.168.5.162
s=session
c=IN IP4 192.168.5.162
t=0 0
m=message 5060 sip sip:victor@add.ultrapower.com.cn

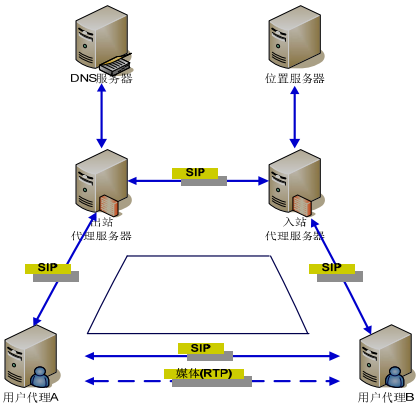
- **Version Number**：协议版本
- **Origin**：所有者/创建者和会话标识符
- **Subject**：会话名称
- **Connection Data**：连接信息
- **Time**：会话活动时间
- **Media(type, port, RTP/AVP Profile)**：媒体名称和传输地址

一个典型的SIP网络结构



- SIP UA或终端构成对话的端点：它发送或接收SIP请求和响应。UA由以下两部分构成：
  - UA Client： 发起请求的主叫方；
  - UA Server：接收、重定向或拒绝请求，代表用户给到来的请求发送响应。

一个典型的SIP网络结构

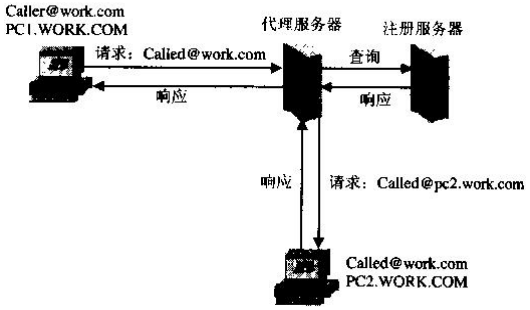


- SIP中间服务器是SIP消息在到达其最终目的地前所经过的逻辑实体，这些中间服务器用于请求路由和重定向。
- 服务器包括：
  - 代理服务器(Proxy Server)
  - 重定向服务器(Redirect Server)
  - 注册服务器(Registrar Server)

SIP实体-代理服务器

- 功能：完成SIP消息的路由
  - 接收SIP终端请求，决定将这些请求传送到何处。
- 当一个SIP终端希望和另一个SIP终端建立多媒体通信会话时：
  - 主叫SIP终端将SIP请求消息发送给它所在控制域的代理服务器
  - 代理服务器通过检测被叫SIP终端的URI，确定是域内转发还是域间转发(功能类似DNS服务器)

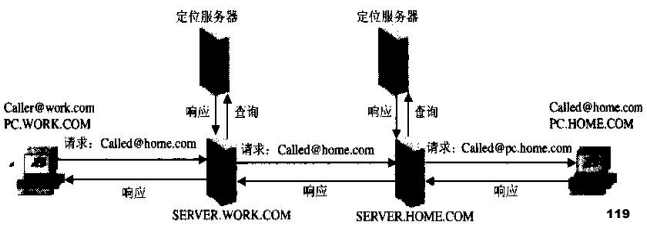
SIP实体-代理服务器



图一 域内转发

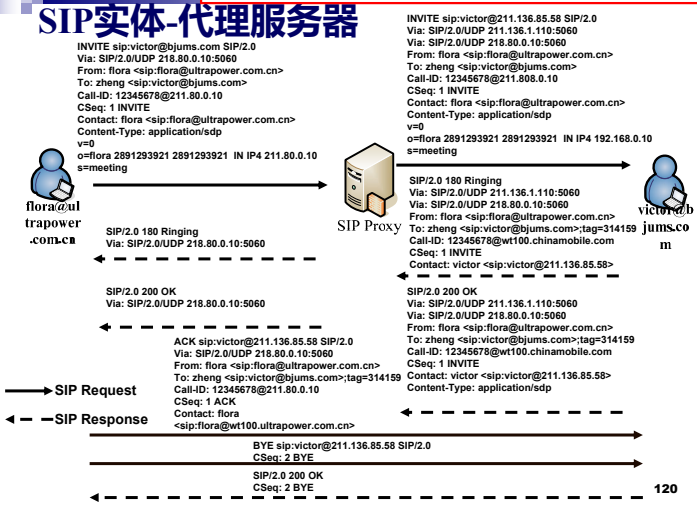
SIP实体-代理服务器

- 如果是域间转发，首先通过定位服务器请求定位服务，获取被叫SIP终端所在控制域的代理服务器的IP地址
- 定位服务器不属于SIP服务器范畴，为SIP重定向服务器和代理服务器获得被叫方可能的位置信息（被叫用户的地址）



篇幅有限，所以省略了向定位服务器的查询服务和另一控制域的代理服务器

SIP实体-代理服务器



SIP实体-重定向服务器(Redirect Server)

- 功能：减少负责路由的代理服务器的负荷
- 实现：只是返回用户有可能出现的位置列表，由用户代理进行用户定位的所有尝试。位置列表将放置在Contact头域中。

响应的3XX类有：

"300" ； Multiple Choices

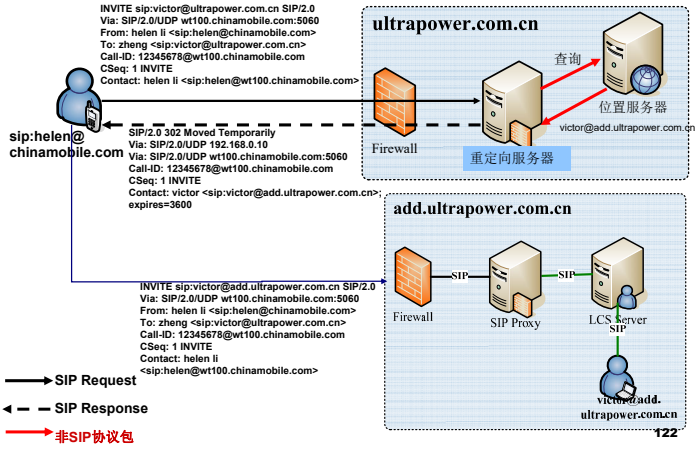
"301" ； Moved Permanently

"302" ； Moved Temporarily

"305" ； Use Proxy

"380" ； Alternative Service

SIP实体-重定向服务器(Redirect Server)



SIP实体-注册服务器(Registrar)

- 目的：接收UA的注册/注销请求，从位置服务器中将UA的地址信息添加/删除，每个SIP终端启动后都需要进行注册。
- SIP的User Mobility:用户通过将自己的AOR与某个主机地址进行明确绑定，使自己可以被联络到，从而使用户移动性成为可能。

SIP实体-注册服务器(Registrar)

注册机制

- 当UA要向注册服务器添加一个地址映射记录时，Contact域包含要增加的联系地址信息，通过Expires头部域或该地址信息的expires 参数来声明该联系地址的生命期。用户可通过一个REGISTER请求消息同时增加多个地址映射记录。

SIP实体-注册服务器(Registrar)

注销机制

- 当UA要删除一个映射记录时，可在Contact域中填写要删除的联系地址信息，并将expires参数置0， 注册服务器收到后就会删除该映射记录。
- 如果将Contact域设为“*”，且Expires头部域设为0，将会删除该用户的所有联系地址映射记录。

SIP实体-注册服务器(Registrar)

创建多个绑定的方法：

- 从每一个设备发送一个REGISTER请求；
- 从同一个设备发送一个与AOR有多个绑定的REGISTER请求。

构造注册请求消息：

Request-URI：包含注册服务器的域名信息；

To：要注册或注销的用户的逻辑地址；

From: 发送注册消息者的地址记录；

Contact: 要注册的联系地址信息。

SIP实体-注册服务器(Registrar)

刷新机制：

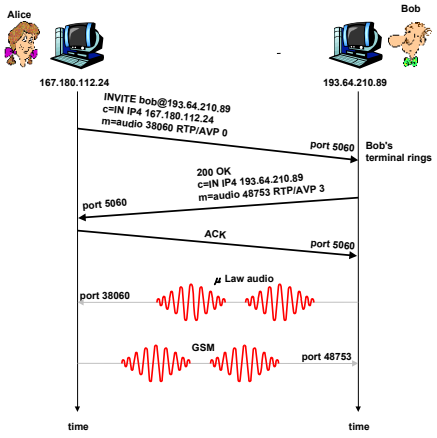
- Contact头域指明了过期参数，默认是1小时。UA应该每隔一段时间后重注册。如果UA没有刷新或明确清除该绑定，则当绑定过期时，Registrar将其直接删除。

SIP实体-注册服务器(Registrar)

SIP服务

- 建立呼叫
  - 允许主叫方通知被叫方它要发起一个呼叫
  - 使主叫方和被叫方能够就媒体类型和编码取得一致
  - 提供结束呼叫的机制
- 确定被叫方当前的IP地址
  - 用户可能分配到动态IP地址
- 呼叫管理
  - 在呼叫期间增加新的媒体流
  - 在呼叫期间改变编码
  - 邀请新的参与者加入
  - 转移和保持呼叫

例子：建立到一个已知IP地址的呼叫



- Alice的SIP invite报文指示了她的端口号&IP地址、Alice首选接收的编码(PCM μ law)、她希望在38060端口接收RTP分组。
- Bob的200 OK SIP响应报文指示了他的端口号、IP地址和他希望接收的编码(GSM)
- SIP报文能够经TCP或UDP发送，这里经 RTP/UDP发送
- 默认的SIP端口号是5060.

例子：建立到一个已知IP地址的呼叫

- 编解码协商:
  - 假定Bob没有 PCM μlaw编解码器.
  - Bob将回答 “606 Not Acceptable Reply”并列出他能使用的编码列表
  - Alice则能发送一个新的INVITE报文，通告一种适当的编码器
- 拒绝该呼叫
  - Bob能拒绝呼叫，回答 “busy,” “gone,” “payment required,” “forbidden”.
- 媒体能够经RTP或某种其他协议发送

SIP 报文的例子

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

注意：类似HTTP报文语法

- 这里我们不知道Bob的IP地址，所以中间的SIP服务器将是必要的
- Alice将用SIP默认端口5060发送和接收SIP报文
- Alice 在Via: 首部定义，SIP客户机经UDP发送和接收SIP报文
- sdp =会话描述协议 (session description protocol)
- Call-ID对每个呼叫都是独特的

例子3：SIP注册和呼叫流程分析

点此观看分析

与H.323的比较

- H.323是另一个用于实时、交互的信令协议
- H.323是一个用于多媒体会议的完整、垂直综合的协议集合：信令、注册、准入控制、传输和编解码器
- SIP是单一组件。与RTP一到工作，但不强制。能与其他协议和服务结合
- H.323 源于ITU (电话).
- SIP 源于 IETF: 借用了HTTP的许多概念。SIP具有Web特征，而H.323具有电话特征
- SIP使用KISS原则：Keep it simple stupid.

第五章 多媒体网络

- 多媒体网络应用
- 流式存储视频
- IP语音
- 实时会话式应用的协议
- 支持多媒体的网络

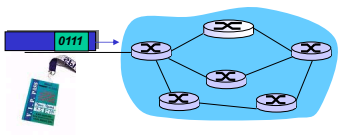
尽力而为的网络

迄今为止：尽最大努力服务

- 公用因特网的路由器同等对待所有分组
- 对时延敏感的音频和视频多媒体流不承诺任何传输质量
  - 解决方案：部署足够的链路容量，使拥塞不会发生，多媒体业务流量无延迟或丢失
    - 面对的问题：在整个网络中提供充足的带宽所带来的高成本；对于如此多的流量，需要确定多少带宽才是“足够”的？；如何定制网络以保证满足网络端点之间的应用及性能?

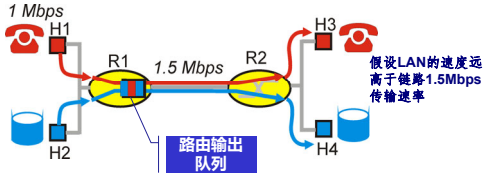
提供多种类型服务的网络

- 具有Qos保证的下一代因特网
  - 区分服务：将流量分类，网络根据不同类别的流量区别对待（类比：VIP服务与常规服务）
  - 每连接服务质量保证Qos：资源预约和呼叫准入



服务场景1：HTTP与VoIP混合

- 例子： 1Mbps VOIP，HTTP共享1.5 Mbps 链路
  - HTTP的突发块能够导致路由器拥塞，引起音频丢包
  - 需要给音频比HTTP更高的优先权

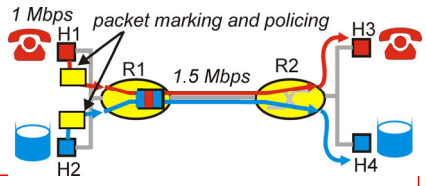


原则1

对分组分类使得路由器可以区分属于不同类别流量的分组；新的路由器策略相应地处理这些分组

QoS保证原则

- 如果应用程序行为不端，例如：音频应用和HTTP共享1.5 Mbps 链路，但音频应用以>1Mbps(宣称的传输速率)的速率发送分组
  - 后果：HTTP分组无法在链路上获得任何服务支持
  - 监管：强迫发送源遵守预先分配的带宽
  - 在网络边缘标记并监管数据流：

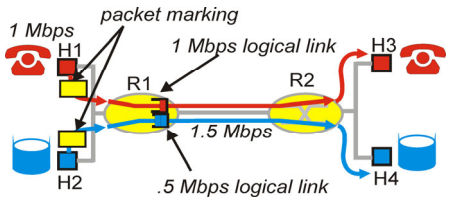


原则 2

希望在流之间提供一定程度的隔离，以便一个流不会受到另一个异常流的负面影响

QoS保证原则(续)

- 为不同类型的流分配固定(不可共享)的带宽，可能造成带宽的低效使用

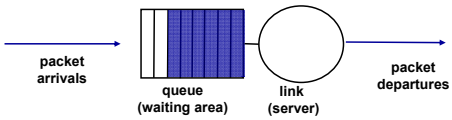


原则 3

间提供隔离时，希望尽可能有效地使用资源（例如链路带宽和缓冲区）

调度与监管机制

- 调度：选择在链路上发送的下一个分组
- FIFO (先进先出) 调度：按到达队列的次序发送
  - 丢弃策略：如果分组到达满队列：谁将被丢弃？
    - 尾丢弃：丢弃最新到达的分组
    - 优先权：基于优先权丢弃/移动
    - 随机：随机地丢弃/移动



先进先出的最大缺点就是不能区分时间敏感分组和一般数据分组，并且也不公平。

FIFO的随机早丢弃策略

- 随机早丢弃（Random Early Drop）算法在路由器分组队列长度达到一定值时，就按照一定的概率将新到来的分组丢弃。
- 路由器采用平均队列长度而非当前的队列长度来计算分组的丢弃概率，从而允许短期的突发负载。

FIFO的随机早丢弃策略

- RED算法计算队列平均长度AvgLen：

$$AvgLen = (1 - Weight) * AvgLen + Weight * SampleLen \quad 0 < Weight < 1$$

SampleLen：当前的队列长度

Weight：加权系数

- 平均队列长度AvgLen决定了路由器队列容纳突发性数据流的程度。
- 分组随机丢弃概率决定了在当前拥塞程度下，路由器丢弃分组的频率。

FIFO的随机早丢弃策略

- 为了计算分组丢弃的概率，引入了两个队列长度阈值，分别是：

最小队列阈值(MinThreshold)和最大队列阈值(MaxThreshold)。

  - 如果AvgLen≤MinThreshold，说明路由器比较空闲，因此丢弃分组的概率为0，将该分组加入队列中。
  - 如果MinThreshold<AvgLen<MaxThreshold，说明路由器已经有一些分组在队列中，这时按公式计算出分组随机丢弃概率，随机丢弃一些分组，使丢弃分组对应的源可以降低发送的速率。
  - 如果AvgLen≥MaxThreshold，说明缓冲区已经比较满了，这时丢弃分组的概率为1，即丢弃所有新到来的分组。

分组随机丢弃概率

- 分组丢弃概率的计算公式如下:
  - 当MaxThreshold<AvgLen<MinThreshold时，counter是从上次分组丢弃到当前时刻为止，放入队列中的新的没有被丢弃的分组的个数。 :

$$TempP = \frac{AvgLen - MinThreshold}{MaxThreshold - MinThreshold} * MaxP$$

最终丢弃概率：$P = \frac{TempP}{1 - count * TempP}$

FIFO的随机早丢弃策略

- 要想有效实现RED策略，还必须对RED固定参数做适当设置：
  - $Weight \geq 0.001$
  - $MaxThreshold \geq 2 * MinThreshold$
  - $MinThreshold \geq Max$（数据流允许的最大突发长度）
  - $MaxP$：丢弃分组概率的最大值
- RED有助于减少链路中的不公平现象，通过早丢弃来控制队列长度的增长，从而减少了大多数分组的延迟。1998年，RED被推荐为路由器端拥塞避免机制的首要方案。

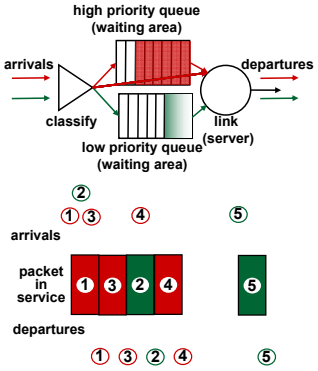
随机早丢弃RED策略总结

- RED策略的缺陷:
  - 平均队列长度随着拥塞的程度与参数的设置而改变
  - 在实际应用中，RED参数都是根据经验进行配置，关于如何配置这些参数，并没有统一意见。

调度策略：优先权

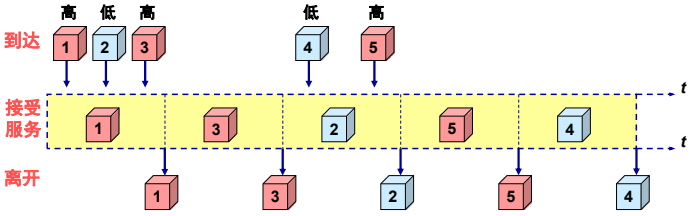
优先权调度: 传输队列中具有最高优先权的分组

- 具有不同优先权的多个类别的分组
  - 类别可能取决于：首部标记字段，如：IPV4的Tos字段；或其他首部信息，如：IP源/目的、端口号等



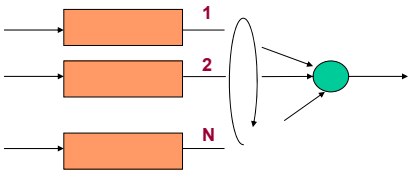
调度策略：优先权

- 缺点：高优先级队列中总有分组时，低优先级队列中的分组将长期得不到服务
- 解决方案：采用轮转或公平排队策略



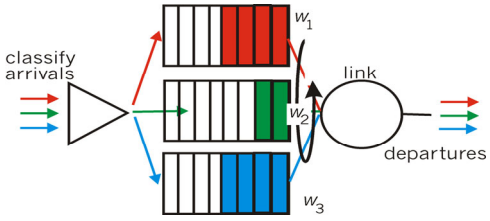
调度策略：轮转

- 对每种类别的分组流设置一个队列，然后轮流使每一个队列一次只能发送一个分组，对于空的队列就跳过去。
- 缺点：长分组得到更多的服务时间；没有对不同优先级的分组提供不同的服务时间。



调度策略：加权公平排队WFQ

- 根据各类别分组的不同优先级，为每类分组队列提供不同的服务时间。



调度策略：加权公平排队WFQ

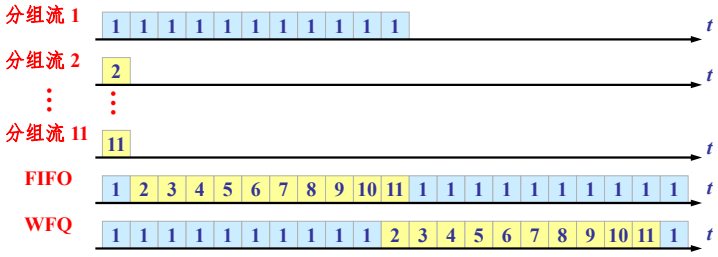
- 分组到达后就将分组进行分类，然后送交与其类别对应的队列。队列按顺序依次将队首的分组发送到链路。遇到队列空就跳过去。
- 给队列 i 指派一个权重 w_i。队列 i 得到的平均服务时间为 $w_i/(\Sigma w_j)$，这里Σw_j是对所有的非空队列的权重求和。
- 队列 i 将得到的有保证的带宽 R_i 应为：

$$R_i = \frac{R \times w_i}{\Sigma w_j}$$

WFQ 与 FIFO 的比较

- 假设分配给分组流 1的权重是0.5，分配给其他10个分组的权重是0.05

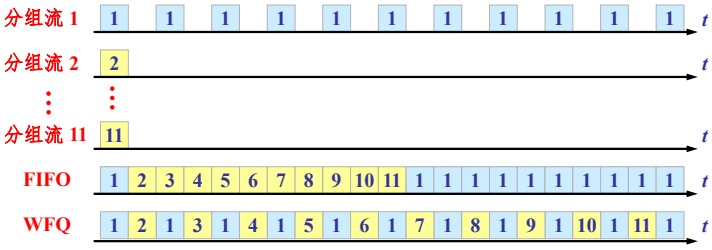
(a) 分组流 1 的分组连续输入



WFQ 与 FIFO 的比较

- 假设分配给分组流 1的权重是0.5，分配给其他10个分组的权重是0.05

(b) 分组流 1 的分组断续输入

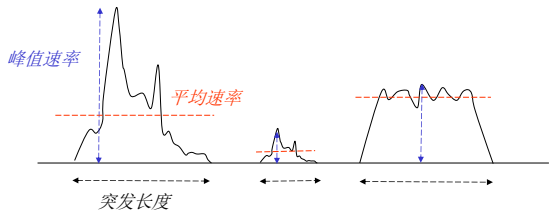


监管机制——三种常用准则:

目的: 限制流量不超过宣称的参数,，是一种重要的QoS机制.

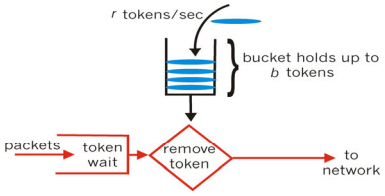
- (长期) 平均速率: 每单位时间能发送多少分组 (在一个长时间范围内)
  - 至关重要的问题：监管平均速率的时间间隔多长? 每秒100分组或每分钟 6000 分组具有相同的平均值!但前者较后者的发送源受到更严格的约束!
- 峰值速率: 限制在一个较短时间内能够发送的最大分组数，如每分钟 6000个 分组 (ppm)平均速率监管一个数据流；但限制该流的峰值速率为 1500 ppm
- (最大的) 突发长度: 极短时间间隔内，能够连续发送的分组最大数量(没有中间的空闲)

监管机制——三种常用准则:



监管机制：实施

令牌桶: 令牌产生速率r用于限制分组能够进入网络的长期平均速率



- 如果桶中少于b个令牌，新产生的令牌被加入该桶，产生令牌的速率是r 令牌/sec；否则忽略新产生的令牌，令牌桶保持具有b个令牌的满状态
- 分组必须获得一个令牌才能向网络传输，显然，经长度t时间间隔：许可的分组数量≤(r t + b).

令牌桶算法中突发时间长度的计算

- 令牌桶允许突发的通信量，但突发分组长度是有限的。
  - 假设突发时间长度为S秒，令牌桶的容量为C字节，令牌产生速率为ρ字节/秒，分组最大输出速率为M字节/秒。
  - 输出的突发数据流有一个最大值：$C + \rho S$ 字节。
  - 又因为在S秒内以最高速率输出的突发数据字节为MS。因此我们有： $C + \rho S = MS$
  - 解这个等式，得到： $S = C/(M - \rho)$

令牌桶算法示例

- 令牌桶容量b = 250KB
- 令牌到达速率r = 2MBps
- 网络最大传输率M = 25MBps

假设：突发数据长为1MB，到达时令牌桶已满。

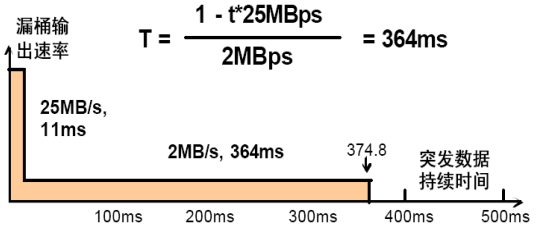
设：突发时间为t；令牌到达速率r；
突发数据最大输出为b＋r · t
b＋r · t = M · t

$$t = \frac{b}{M - r}$$

令牌桶算法示例

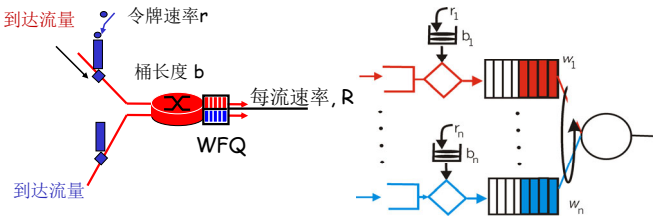
当b = 250K，M = 25MBps，r = 2MBps；
t = 250K/230000K ≈ 10.8ms

- 在t秒内以25MBps的速率突发一部分
- 其余的数据只能以2MBps的速率发送



监管机制：实施

- 令牌桶，WFQ结合提供了时延的确保上界， 即*QoS 保证*!



每个流i保证收到至少等于$R \times w_i \div (\sum w_j)$的共享链路带宽，
其中R是以分组/秒为单位的链路传输速率，当以WFQ方式等待服务时，
如果令牌桶最初都是满的，且有b_i个分组的突发到达流i的漏桶监管器
则分组经受的最大时延$D_{\max} = \dfrac{b_i}{R \times w_i \div (\sum w_j)}$